JBI staat voor Java Business Integration, op dit moment ook bekend als JSR 208 (zie [1]). JBI kan ook gezien worden als een poging tot (Java) standaardisatie in de ESB-hoek (Enterprise Service Bus). Aan de hand van het belang van business integratie zal JBI in dit artikel gepositioneerd worden naast andere manieren van integratie. Verder zal dit artikel de rol van ESB verklaren en zo JBI in perspectief plaatsen.

JBI in perspectief

Standaardisering van services voor integratie

Diverse business redenen en economische ontwikkelingen zorgen ervoor dat integratie weer volop in de belangstelling staat. De kosten van systeem integratie zijn hoog en de looptijd van integratieprojecten is lang. Te lang, als we in ogenschouw nemen dat snelheid geboden is om de concurrent voor te blijven bij het aanbieden van nieuwe diensten, waarbij integratie steeds vaker een belangrijke rol speelt. JBI moet een alternatief bieden voor proprietary en de vaak ad-hoc integratie van systemen door een op standaarden gebaseerde integratie oplossing.

NOODZAAK TOT INTEGRATIE Onder druk van allerlei economische ontwikkelingen zien we de aandacht op zowel business als technologisch vlak voor systeem integratie toenemen. Bedrijven bevinden zich in een steeds zwaardere concurrentieslag en zijn genoodzaakt door middel van bijvoorbeeld fusies, optimalisaties van bedrijfsprocessen of het leveren van nieuwe diensten op basis van bestaande systemen zichzelf te handhaven. Bij fusies van bijvoorbeeld verzekeraars worden bijvoorbeeld optimalisaties bereikt door de backoffice processen te stroomlijnen en te integreren. Een goed voorbeeld van optimalisatie van bedrijfsprocessen is Dell. Door de pc's rechtstreeks aan eindklanten aan te bieden, verdwijnen kosten van tussenschakels als groothandel en detailhandel met als gevolg een lagere consumentenprijs. Het flexibel kunnen verschuiven, samenvoegen of splitsen van onderdelen van de bedrijfsvoering stelt ook hoge eisen aan de integratie van de onderliggende gebruikte systemen.

De standaarddiensten of producten waarmee bedrijven groot zijn geworden, hebben vaak niet meer de toegevoegde waarde om bestaande klanten te behouden of nieuwe klanten te verkrijgen. Snel kunnen integreren en hiermee inspelen op nieuwe eisen is dan ook geen vraag meer maar een *must*. Wat ook de integratiebehoefte vergroot is de "pervasive computing" gedachte: de trend om overal informatie beschikbaar te hebben waardoor processen (en dus systemen) meer naar elkaar toegroeien. Terwijl er steeds hogere eisen worden gesteld aan de flexibiliteit van de systemen die we moeten bouwen, is er minder budget beschikbaar voor nieuwbouw. Onderzoek wijst uit dat nog steeds zeventig tot tachtig procent van de totale IT budgetten wordt uitgegeven aan onderhoud van de bestaande grote enterprise systemen zoals CRM, ERP en zelfgebouwde 'eilanden'. Hierdoor blijft er weinig geld over voor het ontwikkelen van nieuwe complexe systemen en diensten. Onder druk van deze economische bewegingen groeit de noodzaak nieuwe diensten te implementeren door integratie van bestaande systemen. Hierbij zien we ESB en straks wellicht ook JBI een steeds belangrijkere rol spelen omdat de huidige manier van integratie niet voldoet.

POINT-TO-POINT Het integreren van systemen is niet nieuw, dat doen we al jaren. Er zijn verschillende manieren om systemen te integreren. Via JDBC is het mogelijk de database van een ander systeem te benaderen. Ook is het mogelijk een XML-data uitwisseling op te zetten tussen systemen waarbij bijvoorbeeld eens per etmaal een file klaargezet wordt. Dit laatste, ook wel bekend als Extract, Transform en Load (ETL), is een populaire manier van integratie waarbij met behulp van batch processen, scripts, handmatige acties en FTP, data van het ene systeem naar het andere wordt getransporteerd.

De genoemde oplossingen zijn misschien geschikt voor een zogenaamde point-to-point koppeling maar er

zitten een aantal grote nadelen aan. Betrouwbaarheid, het gaat tenslotte om bedrijfskritische data, en het goed afhandelen van fouten wordt amper afgedekt. Het zijn specifieke oplossingen, waarbij koppelingscode moet worden gebouwd waarin business logica herhaald gaat worden in het te koppelen systeem. Verder is de aard van de koppeling vaak synchroon, wat betekent dat het koppelen van meer dan twee systemen eigenlijk onmogelijk is vanwege de runtime afhankelijkheden, die hierdoor gecreëerd worden. Ook wordt bij enterprise integratie de business logica overal in koppelingscode verspreid om nog maar te zwijgen van de daarmee gepaard gaande security, transactie- en infrastructuur problemen. Eigenlijk kunnen we stellen dat waar we aan de bovenkant flexibiliteit nastreven, er in de praktijk zoveel afhankelijkheid ontstaat dat we uiteindelijk juist minder flexibel zijn geworden. We zullen straks zien hoe met ESB deze problemen aangepakt kunnen worden.

Er zijn nog andere manieren van integratie zoals de Integration Brokers (Hub & Spoke) en het gebruik van Message Oriented Middleware. Ook worden in dit rijtje soms applicatieservers genoemd, maar dit is geen echte integratietechnologie omdat het vereist, dat alle te integreren componenten op dezelfde technologie gebaseerd zijn - en dat is bij integratie van heterogene systemen nu juist niet het geval. Een centrale hub bij hub & spoke reduceert het aantal point-to-point verbindingen en beslissingen over bijvoorbeeld routering van data worden centraal genomen. Alhoewel technisch bruikbaar wordt, vanuit de organisatie gezien, centrale controle vaak als te star ervaren. Doordat integratie juist ook over cultuur, politiek en autonomie gaat, is een flexibeler model een vereiste. Daarnaast heeft een centrale oplossing negatieve effecten op schaalbaarheid en robuustheid. Message Oriented Middleware lost weliswaar een aantal problemen op, maar is geen complete integratie oplossing op zich. Wel is het een van de belangrijke onderdelen van ESB.

Java Connector Architecture

Vanuit de gedachte dat er veel geld zit in ERP, legacy en transactie processing systemen, heeft Sun de Java Connector Architecture (JCA) ontwikkeld. JCA biedt een oplossing om heterogene systemen met de Java wereld, en dan met name J2EE, te integreren. Hierbij worden aspecten als security, transactie en connectie management geregeld. Dat is een stap in de goede richting voor business integratie; redelijk gesloten systemen kunnen we nu immers openstellen voor integratie met bestaande en nieuwe ontwikkelingen. De koppeling die gerealiseerd wordt, is alleen 'slechts' gebaseerd op een adapter. Hoewel we het Enterprise Information System (EIS) op het juiste niveau binnengaan, gebruik makend van de daar aanwezige business logica, moeten we nog steeds de koppelingscode zelf implementeren. Eigenlijk communiceren we dus met een EIS op een RPC (Remote Procedure Call) -achtige manier.

ESB - Enterprise Service Bus Een Enterprise Service Bus architectuur kan het best uitgelegd worden als een centrale bus waarop service containers ingeplugd kunnen worden. Deze service containers (een container draait als een proces op het OS) maken selectieve deployment mogelijk en bevatten ondersteuning voor onder meer security, logging, transacties en management.
Een container biedt services aan op de bus door gebruik te maken van ESB endpoints. Dit zijn adapters voor het aansluiten van clients op basis van bijvoorbeeld JCA, JMS, EJB, HTTP, FTP maar ook adapters die niet gebaseerd zijn op een standaard protocol. Er zijn vier belangrijke pijlers onder ESB:

1. *Message Oriented Middleware (MOM)*
   Maakt het mogelijk om asynchroon berichten door de enterprise te sturen. Belangrijk om systemen autonoom te houden is ontkoppelen van zender en ontvanger, een typisch kenmerk van MOM. Voor koppelingen tussen bedrijfskritische systemen is het van belang te communiceren op basis van een asynchroon model met behoud van 'delivery guarantee'. Een systeem kan vaak niet gaan staan wachten totdat het andere systeem beschikbaar is.

2. *Webservices (SOA)*
   Door de te integreren applicatie of dienst als service aan te bieden op de bus, wordt de werkelijke implementatie volledig afgeschermd. Andere services op de bus zien alleen een "abstract end point": een logische abstractie van de service. De integratie-architect kan de procesflow definiëren in termen van deze end points en hoeft op dat niveau dus niets te weten van de onderliggende implementatie of fysieke locatie.

3. *XML gebaseerde data uitwisseling*
   Als we systemen koppelen hebben we te maken met verschillende dataformaten. ESB biedt de mogelijkheid hiervoor transformatieservices in te zetten. Deze services verzorgen de omzetting van XML-data, zodat de output van een systeem geschikt wordt als input voor een ander systeem. Naast transformatie is XML doordat je 'erin kan kijken' zeer geschikt voor Content Based Routing (zie ook punt 4) maar ook logging en auditing is eenvoudiger omdat de data zelfbeschrijvend is.

4. *Routing intelligentie*
   Wat we eigenlijk zien bij enterprise integratie is dat business proces logica over systeemgrenzen (en soms ook bedrijfsgrenzen) heen gaat. Dit vraagt om een centraal model van het business proces waar de stappen decentraal uitgevoerd van kunnen worden. Met routering services kan de business flow, bijvoorbeeld

gebaseerd op BPEL, beschreven worden. Content Based Routing maakt het mogelijk om bijvoorbeeld op basis van XPath routeringsbeslissingen te nemen op basis van de inhoud van een bericht.

Belangrijk is, dat bij ESB alle koppelingskennis via aparte services wordt gemodelleerd of beter gezegd *geconfigureerd*. Dit maakt het mogelijk de bestaande systemen onaangetast te laten en de integratie- en proceslogica samen met de benodigde datatransformaties, routing en transformatie services 'op de bus' te configureren. Dus in plaats van de koppelingscode uit te programmeren en deze logica te beleggen bij de systemen zelf, wordt dit nu onderkend als een apart aspect en bijvoorbeeld via een WS-BPEL service geconfigureerd. Complete ESB-producten zijn op de markt sinds mei 2003. Leveranciers van op ESB gebaseerde producten zijn bijvoorbeeld Sonic, IBM, BEA, Fiorano, Iona en Kenamea. Voor meer details over ESB zie [2].

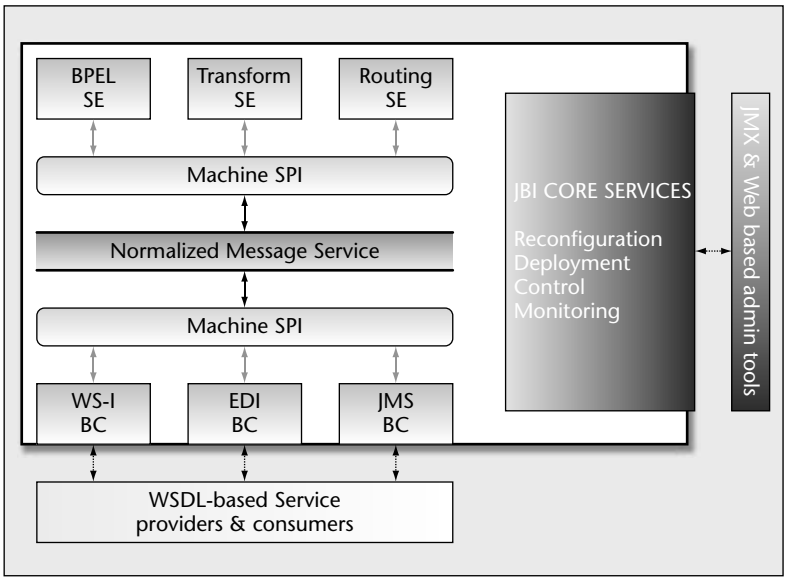
JBI - JAVA BUSINESS INTEGRATION Bij het toepassen van ESB hebben we naast de te integreren services ook een aantal standaardservices nodig om het geheel te laten werken. Een voorbeeld hiervan is een routing service, die de route van messages bepaalt of een transformatie service die een message kan omzetten van het ene formaat naar het andere. Het doel van JBI is nu om de omgeving van dit soort services te standaardiseren. Hierdoor wordt het in de toekomst mogelijk standaardservices van verschillende leveranciers te combineren, of met de tijd over te schakelen op standaard services van een andere partij. Binnen JBI noemt men dit soort standaardservices overigens "Service Engines" (SE).

Het standaardiseren van de omgeving van een SE betekent natuurlijk met name het vastleggen van het interface waarlangs de SE met de buitenwereld communiceert. In JBI wordt dit Machine / Engine SPI (Service Programming Interface) genoemd. In tegenstelling tot wat je misschien zou verwachten, communiceert dit niet direct met de buitenwereld. Daar zitten nog twee schakels tussen (zie Figuur 1) een Normalized Message Service (NMS) en een Binding Component (BC). Om met de laatste te beginnen: de BC zorgt voor de vertaling van het protocol dat de buitenwereld spreekt - bijvoorbeeld SOAP - naar het protocol wat op de NMS gedefinieerd is en vice versa. Hiermee wordt protocolonafhankelijkheid bereikt: spreekt de buitenwereld een ander protocol, bijvoorbeeld JMS, dan is het voldoende om een ander Binding Component in te zetten. Overigens wordt er binnen JBI helemaal niks vastgelegd over de inhoud van de SE's. Het is de bedoeling dat deze SE's, bijvoorbeeld een routing engine, worden gedefinieerd in een aparte standaard (zie [3]). Het enige wat JBI vastlegt met betrekking tot dit soort services, is op welke manier ze (via de NMS) communiceren, de inhoud van communicatie valt buiten deze standaard.

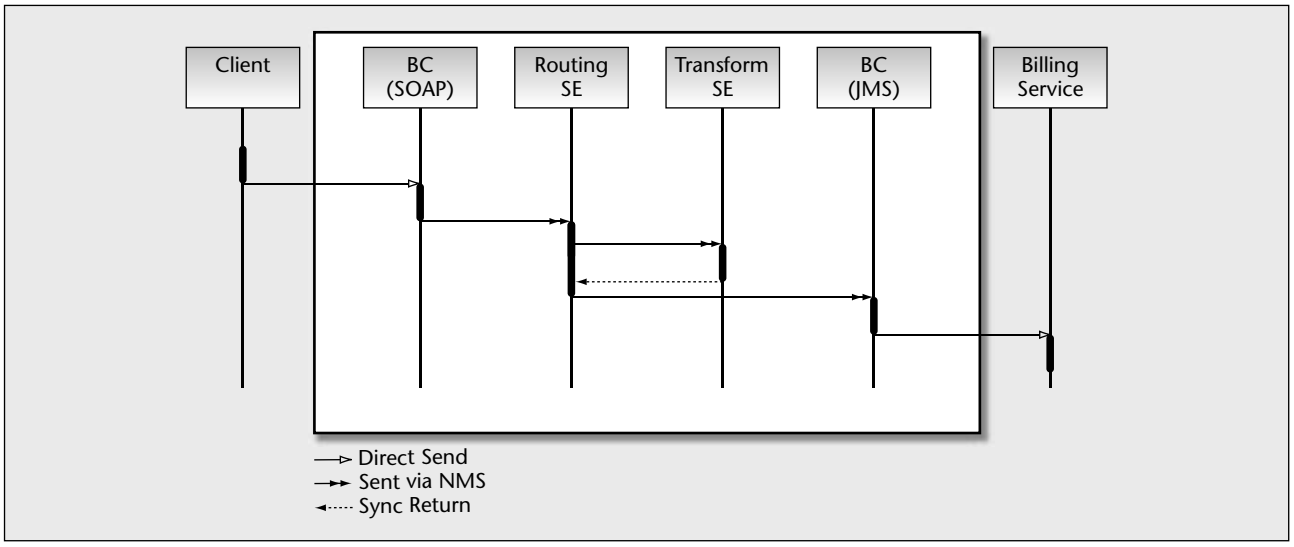
CORE SERVICES De NMS definieert een standaard communicatiekanaal binnen de JBI container. Binnen een JBI container kunnen meerdere SE's actief zijn en de onderlinge communicatie verloopt rechtstreeks via deze interne bus. Als we dit plaatsen in de context van ESB, dan zie we dat een JBI container in feite een ESB container is. Weliswaar een speciale, omdat binnen deze container alles gestandaardiseerd is en volledig op Java gebaseerd, maar vanuit de ESB gezien is het slechts een ESB container die een of meer services host. JBI moet dan ook niet gezien worden als een vervanging van ESB, maar als een Java container voor het hosten van belangrijke ESB services. In Figuur 2 is te zien hoe een bericht uit de ESB naar een service in een JBI container wordt gestuurd.

Tenslotte biedt de JBI container ook nog een aantal JBI core services. Hieronder valt onder meer lifecycle management van de componenten die in deze container draaien. Om de container met zijn core services te configureren en te managen wordt gebruik gemaakt van JMX. De status van de JBI (JSR 208) is sinds 8 oktober jl. "Early Draft Review". Deze review periode loopt tot 22 november. Over hoe lang het nog gaat duren voor er een "Public Review" of een "Final Draft" is, is weinig te zeggen. Feit is wel dat deze eerste fase behoorlijk lang geduurd heeft en dat de "Early Draft" zoals die er nu ligt nog opvallend veel leemtes vertoont. Ook wordt er in deze draft herhaaldelijk om input van het publiek gevraagd. Op grond van deze signalen kan het nog wel eens een tijdje gaan duren voor de specificatie er feitelijk ligt.

CONCLUSIE Enterprise integratie wordt steeds belangrijker, en is het stadium van ad hoc koppelingen



FIGUUR 1. JBI container



FIGUUR 2. Een bericht uit de ESB wordt naar een service in een JBI-container gestuurd.

ontgroeid. ESB en ook JBI bieden handvatten om integratieproblematiek op een structurele manier aan te pakken. Hoe belangrijk is het koppelen van systemen voor jouw organisatie? Gaat het om kernsystemen van de organisatie waarop steeds nieuwe diensten aangeboden worden, dan is het wellicht verstandig om in elk geval naar ESB te gaan kijken. Het is geen silver bullet, maar de rationale erachter geeft veel integratie-inzicht. En met de komst van JBI kan vendor lock-in met ESB voorkomen worden door vendors te kiezen die JBI ondersteunen. Hoe dan ook is het verstandig om Enterprise business integratie serieus aandacht te geven op het niveau van software architectuur.

Referenties

[1] *Informatie over JSR 208: JBI*
http://www.jcp.org/en/jsr/detail?id=208

[2] *Enterprise Service Bus* - David Chapell
O'Reilly, juni **2004**, ISBN 0-596-00675-6

[3] *Informatie over JSR 207: Process Definition for Java*
http://www.jcp.org/en/jsr/detail?id=207

Jeroen Bouvrie en Peter Doornbosch zijn beiden werkzaam bij Luminis

PATCHES Patches PATCHES Patches PATCHES Patches PATCHES

Uitbreiding test- en analysemogelijkheden van Compuware Vantage software

Compuware Vantage verkleint de kans op verrassingen bij de implementatie van bedrijfskritische applicaties. De Vantage software kan bedrijven vanaf nu ook in de pre-productiefase een realistische weergave geven van de beschikbaarheid en prestatie van applicaties. De meest voorkomende problemen kort na de implementatie kunnen hiermee vermeden worden. Vantage bekijkt de situatie vanuit het perspectief van de eindgebruiker. Door Vantage reeds in te zetten in de pre-productiefase kunnen bedrijven de prestatie van bedrijfsapplicaties op alle punten in het netwerk simuleren alvorens ze de toepassing in gebruik nemen. Met uitgebreide voorspellende analyses van Vantage kunnen bedrijven de impact van veranderingen in de IT-omgeving vooraf bepalen. Nieuw zijn de 'what if' scenario's waarmee het effect van mogelijke verbeteringen vooraf is vast te stellen.

Zo visualiseert Vantage de effecten op applicatiegedrag, netwerk bandbreedte, latency of server responsetijden. Vantage rapporteert door middel van duidelijke schema's en tabellen waarmee de test- en analyseresultaten ook inzichtelijk zijn voor het management. De verbeterde integratie tussen Compuware's Vantage en Compuware's QACenter zorgt voor meer prestatietest-mogelijkheden, waardoor er vooraf uitspraken gedaan kunnen worden over de prestaties van bedrijfskritische applicaties in de praktijk.

Zo kan het verwachte aantal gebruikers vooraf gesimuleerd worden en kan worden vastgesteld hoe de applicatie met de bestaande infrastructuur om zal gaan. Ook kan worden bepaald wat de impact van de applicatie op de servers is, van belang voor webservices. In Nederland is Compuware vertegenwoordigd met een Nederlands verkoopkantoor, het Europese hoofdkantoor en het Amsterdam Development Center, waar 200 ontwikkelaars werkzaam zijn. In Amsterdam worden onder andere Compuware's producten als OptimalJ en Uniface ontwikkeld.

Meer informatie vindt u op www.compuware.nl.